

construct a parse tree :-

$$S \rightarrow S(S)S$$

It

string :- $((\lambda))$

so,

$$S \underset{\pi m}{=} S(S) \underset{\pi m}{S}$$

$$\underset{\pi m}{\Rightarrow} S(S) \underset{\pi m}{E}$$

$$\underset{\pi m}{\Rightarrow} S(S)$$

$$\underset{\pi m}{\Rightarrow} S(S(S(S)S))$$

$$\underset{\pi m}{\Rightarrow} S(S(S(S(S)S)))$$

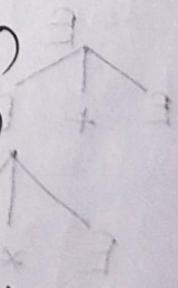
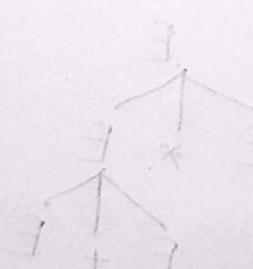
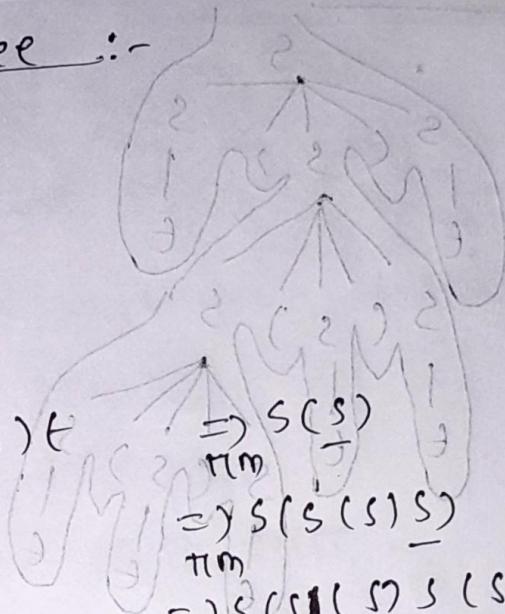
$$\underset{\pi m}{\Rightarrow} S(S(S(S(S))))$$

$$\underset{\pi m}{\Rightarrow} S(S(S(S)))$$

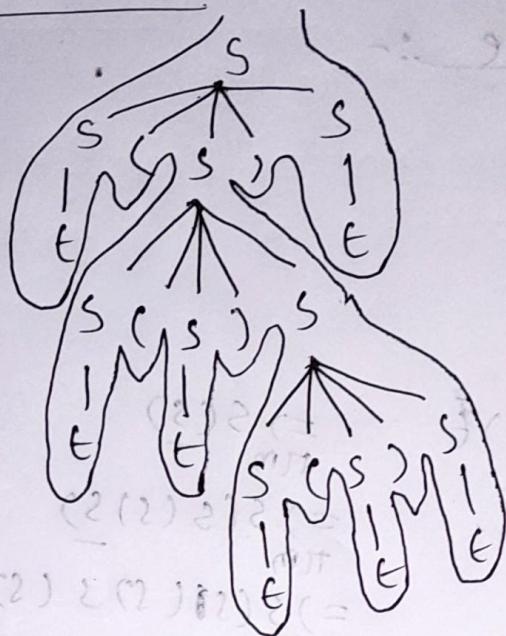
$$\underset{\pi m}{\Rightarrow} S((S)))$$

$$\underset{\pi m}{\Rightarrow} E((S)))$$

$$\underset{\pi m}{\Rightarrow} ((S)))$$



so parse tree:-



~~str!~~: f

$\Rightarrow (1)(2)(3)(4) \cancel{2} \in \mathbb{Z}$

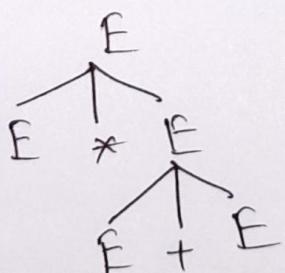
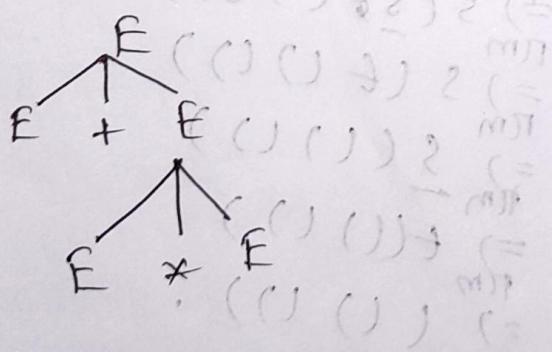
$$E \rightarrow E + E$$

$1E * E(2) id \rightarrow 21315$

1 (1d) (2) ~~82~~) 2 (=

Tree-1:

Tree - 2



→ if any grammar generates 2 different parse tree then it is called ambiguous grammar.

=> You cannot use an ambiguous grammar because it confuses the compiler.

=> Compiler need to be deterministic.

A grammar is called ambiguous if we find:-

- More than one LMD or
- More than one RMD or
- More than one parse tree.

2nd (cc)

10.05.23

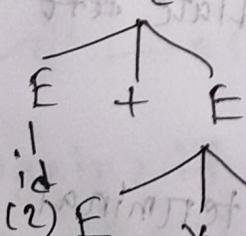
Left recursive grammar \rightarrow Non deterministic grammar.

$$E \rightarrow E + E$$

$$| E * E$$

| id

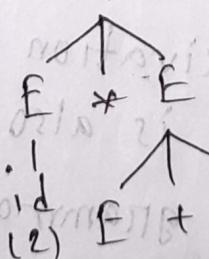
(17)



| E * E

| id

(18)



| E * E

| id

(19)

| id

(20)

| id

(21)

| id

(22)

| id

(23)

| id

(24)

| id

(25)

| id

(26)

| id

(27)

| id

(28)

| id

(29)

| id

(30)

| id

(31)

| id

(32)

| id

(33)

| id

(34)

| id

(35)

| id

(36)

| id

(37)

| id

(38)

| id

(39)

| id

(40)

| id

(41)

| id

(42)

| id

(43)

| id

(44)

| id

(45)

| id

(46)

| id

(47)

| id

(48)

| id

(49)

| id

(50)

| id

(51)

| id

(52)

| id

(53)

| id

(54)

| id

(55)

| id

(56)

| id

(57)

| id

(58)

| id

(59)

| id

(60)

| id

(61)

| id

(62)

| id

(63)

| id

(64)

| id

(65)

| id

(66)

| id

(67)

| id

(68)

| id

(69)

| id

(70)

| id

(71)

| id

(72)

| id

(73)

| id

(74)

| id

(75)

| id

(76)

| id

(77)

| id

(78)

| id

(79)

| id

(80)

| id

(81)

| id

(82)

| id

(83)

| id

(84)

| id

(85)

| id

(86)

| id

(87)

| id

(88)

| id

(89)

| id

(90)

| id

(91)

| id

(92)

| id

(93)

| id

(94)

| id

(95)

| id

(96)

| id

(97)

| id

(98)

| id

(99)

| id

(100)

| id

(101)

| id

(102)

| id

(103)

| id

(104)

| id

(105)

| id

(106)

| id

(107)

| id

(108)

| id

(109)

| id

(110)

| id

(111)

| id

(112)

| id

(113)

| id

(114)

| id

(115)

| id

(116)

| id

(117)

| id

(118)

| id

(119)

| id

(120)

| id

(121)

| id

(122)

| id

(123)

| id

(124)

| id

(125)

| id

(126)

| id

(127)

| id

(128)

| id

(129)

| id

(130)

| id

(131)

| id

(132)

| id

(133)

| id

(134)

| id

(135)

| id

(136)

| id

(137)

| id

(138)

| id

(139)

| id

(140)

| id

(141)

| id

(142)

| id

(143)

| id

(144)

| id

(145)

| id

(146)

| id

(147)

| id

(148)

| id

(149)

| id

(150)

| id

(151)

| id

(152)

| id

(153)

| id

(154)

| id

(155)

| id

(156)

| id

(157)

| id

(158)

| id

(159)

| id

(160)

| id

(161)

| id

(162)

| id

(163)

| id

(164)

| id

(165)

| id

(166)

| id

(167)

| id

(168)

| id

(169)

| id

(170)

| id

(171)

</

Precedence priority :- (+, -, *, /, %)

(setting priority for operators)

- * if there is non-terminal after terminal symbol than it is a infinite or left-recursive grammar.

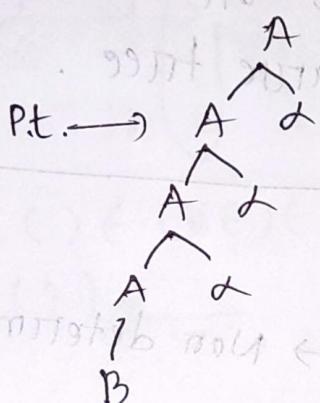
Ex:- $A() \{$

$A();$

$\alpha;$

γ

$A \rightarrow A\alpha | B$



R.E:- $B\alpha^*$

left recursive grammar :-

A grammar has a non-terminal A such as, there is a derivation $A \Rightarrow A\alpha$ for some string α . This process is also known as immediate left recursive grammar.

Problem :-

- Execute the production of a non-terminal symbol without checking the terminal symbol which produce infinite loop on iteration.
- As it executes the production of a non-terminal symbol, so this is the way to

increase probability to make a grammar ambiguous.

(iii) A left recursive grammar makes the iteration with $B\alpha^*$, if the grammar is,

$$A \rightarrow A\alpha | B$$

We have to eliminate the left recursive grammar without changing its iteration $B\alpha^*$. Therefore the grammar should be ton.

if, $A \rightarrow A\alpha | B$ is:-

then,
equ.
$$\begin{cases} A \rightarrow BA' \\ A' \rightarrow \alpha A' | \epsilon \end{cases}$$

$$A \rightarrow \alpha A$$

Ex:-
$$S \xrightarrow{} S \underset{\overline{A}}{\underset{\overline{A}}{\underset{\alpha}{\underset{B}{\mid}}}} \underset{\overline{B}}{\underset{\overline{\alpha}}{\underset{\overline{\epsilon}}{\mid}}} \quad (\text{immediate})$$

so,

$$\begin{aligned} & S \xrightarrow{} 01S' \\ & S' \xrightarrow{} 0S1SS' | \epsilon \end{aligned}$$

Ano

Again, if,

$$A \rightarrow A\alpha_1 | A\alpha_2 | A\alpha_3 | \dots | A\alpha_m | B_1 | B_2 | B_3 | \dots | B_n$$

then,

equ.
$$\begin{cases} A \rightarrow B_1 A' | B_2 A' | B_3 A' | \dots | B_n A' \\ A' \rightarrow \alpha_1 A' | \alpha_2 A' | \alpha_3 A' | \dots | \alpha_m A' | \epsilon \end{cases}$$

Ex:-

$$E \rightarrow E + T \mid E * T \mid a \frac{b}{B_1} \frac{b}{B_2}$$

then,

$$E \rightarrow a E' \mid b E'$$

$$B_1 A \leftarrow A$$

$$E' \rightarrow + T E' \mid * T E' \mid t$$

int. derivat. of given grammar

Non-immediate left recursive grammars

$$S \rightarrow A a \mid b$$

$$A \rightarrow A c \mid S d \mid t$$

$$S \Rightarrow A a$$

$$\Rightarrow \underline{S} d a$$

$$A b c \leftarrow A$$

..... incomplete

$$A g \leftarrow A$$

$$A b \leftarrow A$$

Question:

$$1) L \rightarrow L, SIS$$

$$\frac{10122020}{d} \quad \overline{\overline{A}} \quad \overline{\overline{A}}$$

$$2) E \rightarrow T e \mid a$$

$$T \rightarrow F * T \mid E y d \mid t$$

$$F \rightarrow T l i d$$

$$3) A \rightarrow A B d \mid A a l a$$

$$B \rightarrow B e l b$$

$$4) S \rightarrow A$$

$$A \rightarrow A d l A e l a B l a c$$

$$B \rightarrow b B c \mid f$$

Sol:-

$$\textcircled{1} \quad L \rightarrow L, SIS \quad \begin{matrix} & \\ \overline{A} & \overline{A} \xrightarrow{\alpha} \overline{B} \end{matrix}$$

$$L \rightarrow SL'$$

$$L' \rightarrow SIS' | \epsilon$$

$$\textcircled{1} \quad E \rightarrow Tc1a$$

$$T \rightarrow F * T | Eycde$$

$$F \rightarrow T | id$$

$$1) \quad E \Rightarrow Tc$$

$$\Rightarrow Eycde$$

$$\frac{E}{A} \rightarrow \frac{Eycde}{A \xrightarrow{\alpha}} \frac{1a}{B}$$

$$\therefore E \rightarrow aE'$$

$$E' \rightarrow ydcE' | \epsilon$$

$$\textcircled{11} \quad A \rightarrow ABd \quad \begin{matrix} & \\ \overline{A} & \overline{A} \xrightarrow{\alpha_1} \overline{B} \end{matrix} \quad \begin{matrix} & \\ \overline{A} & \overline{A} \xrightarrow{\alpha_2} \overline{B} \end{matrix} \quad \begin{matrix} & \\ \overline{B} & \overline{B} \xrightarrow{\alpha_m} \end{matrix}$$

$$\frac{B}{B_1} \rightarrow \frac{Be}{B_2} \frac{1b}{B_m}$$

$$\therefore A \rightarrow BA' \quad \begin{matrix} & \\ \overline{B} & \overline{B} \xrightarrow{\alpha_1} \end{matrix} \quad \begin{matrix} & \\ Be & Be \end{matrix} \quad \begin{matrix} & \\ \overline{A'} & \overline{A'} \xrightarrow{\alpha_2} \end{matrix} \quad \begin{matrix} & \\ 1b & 1b \end{matrix}$$

$$2) \quad T \rightarrow T \quad \begin{matrix} & \\ \overline{T} & \overline{T} \xrightarrow{\alpha_1} \end{matrix} \quad \begin{matrix} & \\ T & T \xrightarrow{\alpha_2} \end{matrix} \quad \begin{matrix} & \\ 1a & 1a \end{matrix}$$

$$\therefore T \rightarrow \frac{T}{\alpha_1} \quad \begin{matrix} & \\ \overline{T} & \overline{T} \xrightarrow{\alpha_2} \end{matrix} \quad \begin{matrix} & \\ T & T \xrightarrow{\alpha_3} \end{matrix} \quad \begin{matrix} & \\ 1cycd & 1cycd \end{matrix}$$

$$T' \rightarrow *TT' \quad \begin{matrix} & \\ \overline{T} & \overline{T} \xrightarrow{\alpha_4} \end{matrix} \quad \begin{matrix} & \\ T & T \xrightarrow{\alpha_5} \end{matrix} \quad \begin{matrix} & \\ 1cycd & 1cycd \end{matrix}$$

$$\frac{T}{\alpha_1} \rightarrow \frac{T}{\alpha_2} \quad \begin{matrix} & \\ \overline{T} & \overline{T} \xrightarrow{\alpha_3} \end{matrix} \quad \begin{matrix} & \\ T & T \xrightarrow{\alpha_4} \end{matrix} \quad \begin{matrix} & \\ 1cycd & 1cycd \end{matrix}$$

$$\frac{T}{\alpha_2} \rightarrow \frac{T}{\alpha_3} \quad \begin{matrix} & \\ \overline{T} & \overline{T} \xrightarrow{\alpha_4} \end{matrix} \quad \begin{matrix} & \\ T & T \xrightarrow{\alpha_5} \end{matrix} \quad \begin{matrix} & \\ 1cycd & 1cycd \end{matrix}$$

$$\frac{T}{\alpha_3} \rightarrow \frac{T}{\alpha_4} \quad \begin{matrix} & \\ \overline{T} & \overline{T} \xrightarrow{\alpha_5} \end{matrix} \quad \begin{matrix} & \\ T & T \xrightarrow{\alpha_6} \end{matrix} \quad \begin{matrix} & \\ 1cycd & 1cycd \end{matrix}$$

Bnd (cc)

17.05.23

AC) α

AC);

α

$y \rightsquigarrow BA^*$

$$\boxed{A \rightarrow BA' \\ A' \rightarrow \alpha A' 1 \epsilon}$$

Ex:-

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid id$$

Sol:

$$E \rightarrow E + T \mid T$$

$$E \rightarrow A \mid \overline{A} \mid \alpha \mid \overline{B} \mid T$$

$$E \rightarrow T E' \mid T \epsilon$$

$$E' \rightarrow + T E' \mid \epsilon$$

again,

$$T \rightarrow T * F \mid F$$

$$T \rightarrow F T'$$

$$T' \rightarrow * F T' \mid \epsilon$$

$$x \rightarrow x s b \mid s a \mid b$$

$$s \rightarrow s b \mid x a \mid a$$

$$x \rightarrow x s b \mid b$$

$$x \rightarrow s a \mid b$$

$$x \rightarrow x a a \mid a b$$

Non-determinism

Non-determinism means you have numerous option on a single symbol. consider the following grammar:

$$A \rightarrow \alpha B_1 | \alpha B_2 | \alpha B_3$$

From this grammar you have to derive αB_3 . In order to derive the grammar you need to perform backtracking. In addition, the backtracking is happened because of common prefixes, and it is the concept of non-determinism.

Left Factoring :-

If you have a non-deterministic grammar -

$$A \rightarrow \alpha B_1 | \alpha B_2 | \alpha B_3 | \dots | \alpha$$

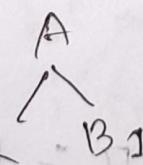
then the deterministic or the determinism of the grammar could have:-

$$A \rightarrow \alpha A' | \alpha$$

$$A' \rightarrow B_1 | B_2 | B_3$$

$$A \rightarrow \alpha A'$$

$$A' \rightarrow B_1 | B_2 | B_3$$



| if there is common-prefix
than it is non-deterministic

Ex:- $s \rightarrow \underline{bss}aa \underline{s}asb \underline{b}sb \underline{a}$

Sol/ $s \rightarrow bss' \underline{l}a$

$s' \rightarrow \underline{s}aa \underline{s}asb \underline{l}b$

Again, as again common prefix found

$$S' \rightarrow S_1 \& S''$$

$$S'' \rightarrow a \& Sb \& \epsilon$$

cc (4th)

18.05 - 23

Elimination of (left factoring) non-deterministic
does not remove the ambiguity :-

stmt \rightarrow if expr then stmt

1 if expr then stmt else stmt

1 a

if \rightarrow i

expr \rightarrow E

then \rightarrow t

stmt \rightarrow s

else \rightarrow e

$S \rightarrow iEt$

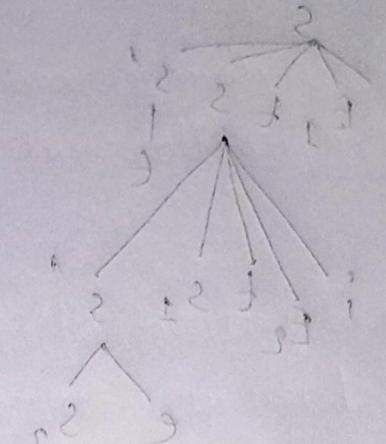
$iEt \rightarrow iEts$

(a)

dangling - else grammar = (After left factoring)

it also can occur ambiguity)

if () {
 }
 if () {
 }
 else {
 }
 S → iEts
 iEtses
 |a

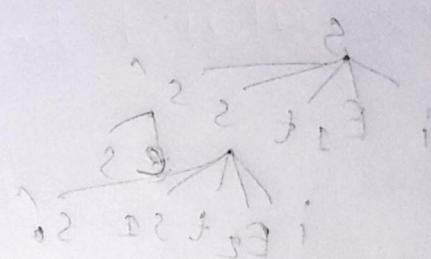


grammar :-

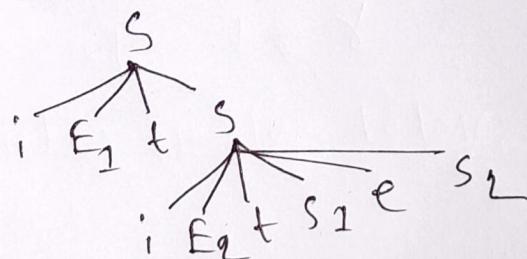
- if E_1 then if E_2 then

s_1 else s_2

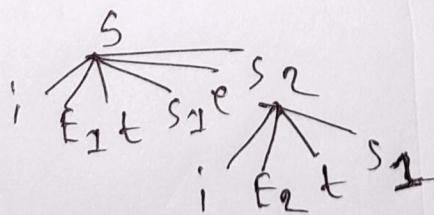
string : iE₁t iE₂t s₁ e s₂



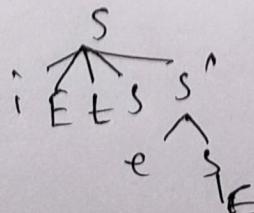
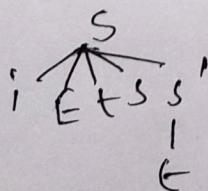
1st production :-



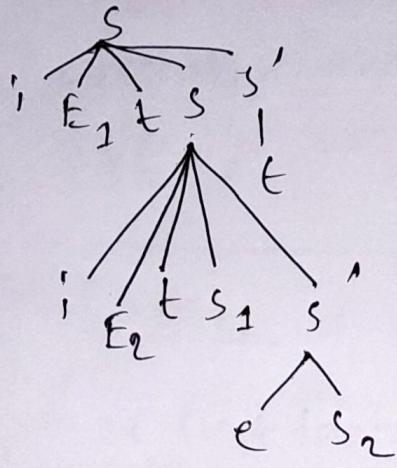
2nd production :-



$S \rightarrow iEtsS'$
 $S' \rightarrow eles1a$



1)



II)

